УДК 004.9.1.2

А.И. Шевченко, М.А. Курилов, Л.П. Сыпченко, А.С. Барашко
Институт информатики и искусственного интеллекта
ГВУЗ «Донецкий национальный технический университет», г. Донецк, Украина
Украина, 83050, г. Донецк, ул. Б. Хмельницкого, 84, *kurilov.ukraine@gmail.com*

Аксиомы программирования и некоторые вопросы Дистанционного Обучения

A.I. Shevchenko, M.A. Kurilov, L.P. Sypchenko, A.S. Barashko
Institute of Informatics and Artificial Intelligence
of Donetsk National Technical University, Donetsk, Ukraine
Ukraine, 83050, c. Donetsk, B. Khmelnitskiy st., 84

Axioms of Programming and Some Issues of Distance Education

А.І. Шевченко, М.О. Курилов, Л.П. Сипченко, А.С. Барашко
Інститут інформатики і штучного інтелекту
ДВНЗ «Донецький національний технічний університет», м. Донецьк, Україна
Україна, 83050, м. Донецьк, пр. Б. Хмельницького, 84

Аксіоми програмування та деякі питання Дистанційного Навчання

В данной работе впервые предложены аксиомы программирования, знание которых в значительной степени способствует быстрой и качественной разработке программного продукта. Рассмотрены некоторые общие методологические принципы и правила ведения программных разработок, в том числе и систем Дистанционного Обучения. Рассмотрены предпосылки интенсивной разработки программ.
Ключевые слова: дистанционное обучение, программирование, проблема общения, информатика, математика, учебный модуль.

In the given work, axioms of programming are offered, that is the novelty. These axioms help in quick and qualitative product development. Some general methodological principles and rules for product development are considered, including systems of Distance Education.Background for program intensive development is considered.
Key words: distance education, programming, communication problem, informatics, mathematics, curricular module.

У даній роботі вперше запропоновані аксіоми програмування, знання яких у значній мірі сприяє швидкій та якісній розробці програмного продукту. Розглянуті деякі загальні методологічні принципи і правила проведення програмних розробок, у тому числі і систем Дистанційного навчання. Розглянуті передумови інтенсивної розробки програм.
Ключові слова: дистанційне навчання, програмування, проблема спілкування, інформатика, математика, навчальний модуль.

Введение

В последнее время во всем цивилизованном мире возрос интерес к новым формам усвоения знаний, в частности к так называемому Дистанционному Обучению (ДО). В рам-

ках отсутствия общепринятого определения этого понятия, обратившись к одному из источников, находим, что ДО − это образование, которое полностью или частично осуществляется с помощью компьютеров, телекоммуникационных технологий и средств [1]. Дидактические аспекты реализации ДО, в зависимости от изучаемой предметной области, очевидно имеют различный характер. Здесь мы будем рассматривать эти аспекты применительно к сфере информационных технологий, точнее к той её составляющей, которая представлена обучению программированию, что, как правило, связано с формальными языковыми системами. Ведь именно эти самые языки (и программирования, и разметки) представляют собой тот самый инструмент, без чего вообще невозможна реализация любого программного обеспечения, в том числе и для ДО.

Целью данной работы является систематизация некоторых аспектов программирования и дистанционного обучения ему, что традиционно находится вне поля зрения специалистов. Предлагаемая статья может быть полезной не только для программистов (как начинающих, так и профессионалов), но и для тех, кто осуществляет руководство ими на уровне разработки самих прикладных программных средств и дистанционных машинных средств по его обучению.

Программирование как математическая наука

Программирование как один из видов интеллектуальной деятельности человека, однозначно можно рассматривать как дисциплину математической природы [2]. Если математические утверждения стремятся к абсолютной точности, то программирование априорно не допускает никакой неопределенности как на уровне спецификации значений используемых данных, так и на уровне управления ходом выполнения самой программы. Обобщенность математических утверждений применительно к программированию отображается фактом функционирования разработанных алгоритмов на базе большого (часто, бесконечного) класса примеров или многообразия входных данных. Сама природа написания программ требует жесткой точности от самого программиста: если инструкции (операторы или команды) были бессмысленны, то и машина будет выдавать бессмыслицу. Настоящее программирование требует огромной тщательности и неизменного таланта и любви к четкой логике и точности; оно трудоемко, и поэтому его следует откладывать до тех пор, пока сам программист не станет максимально уверен в том, что программа, к реализации которой он намерен приступить, это та самая программа, к которой он стремится. За последние несколько десятилетий программно-аппаратный мир информационных технологий колоссально изменился, однако большинство организаций по его разработке используют правило, согласно которому для проекта, рассчитанного на один год, запрещено начинать кодировать не ранее чем через девять месяцев! Окончательный код не более чем залог понимания глубины и сложности тех проблем, решению которых и предназначена собственная разработка программного приложения.

Всякое обучение мы можем рассматривать как усовершенствование прошлого. Применительно к программированию здесь можно отметить приход Объектно-Ориентированного Программирования (ООП) на смену программированию процедурному. Отрицает ли ООП своего предшественника? Однозначно нет. Любой профессионал скажет, что практически любые вычислительные процедуры, реализованные в среде ООП, можно формально описать в рамках «процедурного предка». Такие понятия, как «класс» или «объект» для ООП имеют практически тот же смысл, что и «описание процедуры или функции» и ее вызов в программировании процедурном. Инкапсуляция,

наследование и полиморфизм – все это было и ранее, хотя следует отметить, что при этом размер исходного кода в случае ООП практически всегда будет меньше своего «процедурного» предшественника.

Роль программирования в современном мире

Информационные технологии (ИТ, от англ. information technology, IT), без которых невозможна современная жизнь, представляют собой [1] набор дисциплин и областей деятельности, относящихся к технологиям управления и обработки данных, а также создания данных с применением средств вычислительной техники. В последнее время под информационными технологиями чаще всего понимают компьютерные технологии. В частности, ИТ имеют дело с использованием компьютеров и программного обеспечения для хранения, преобразования, защиты, обработки, передачи и получения информации. Специалистов по компьютерной технике и программированию часто называют ИТ-специалистами. Согласно определению, принятому ЮНЕСКО, ИТ − это комплекс взаимосвязанных научных, технологических, инженерных дисциплин, изучающих методы эффективной организации труда людей, занятых обработкой и хранением информации; вычислительную технику и методы организации и взаимодействия с людьми и производственным оборудованием, их практические приложения, а также связанные со всем этим социальные, экономические и культурные проблемы. Сами ИТ требуют сложной подготовки специалистов, больших первоначальных затрат и наукоемкой техники. Достигнутые результаты в IT среде в значительной степени определяются уровнем профессиональной подготовки тех лиц, которые находятся в данной среде. Для того чтобы оценить саму важность такого уровня подготовки, достаточно обратиться к фундаментальным высказываниям одного из ученых, который стоял у истоков IT, и, по его собственному высказыванию, был самым первым программистом в мире [2].

Перейдем теперь к дистанционному обучению. Избегая бесконечных дискуссий относительно его (ДО) дефиниции, можно однозначно утверждать, что ДО полностью подпадает под юрисдикцию IT. Здесь так же, как и при проектировании любого программного продукта, очень важным является разрешение проблемы общения [3], [4], однако сама проблема здесь должна рассматриваться не только в свете «человек – ЭВМ», но и в аспекте «программист – программист», а также «руководитель программных разработок – программист». Трудно себе представить вариант достижения положительных результатов, когда руководитель программных разработок, не владеющий основами конструирования программных продуктов, не учитывает мнения программистов, входящих в состав группы разработчиков. Ведь программирование можно также рассматривать как особую форму организации мышления и развития человеческой логики. И лучшего инструмента для проведения «гимнастики ума» в мире еще не придумано. Вершиной профессиональной подготовки всякого программиста можно считать его способность адекватно оценить общее время, которое требуется на разработку того или иного программного продукта. Здесь хочется привести один пример, когда в 60-х годах прошлого столетия Институту кибернетики им. В.М. акад. Глушкова были поручены работы по созданию на серии предприятий военно-промышленного комплекса АСУ «Кунцево».

Интересно, что Владимир Ильич Скурихин, ранее участвовавший в создании АСУ на Львовском телевизионном заводе, категорически отказался от участия в создании этой системы: «Я с вилкой на медведя не хожу». Он понимал, что многономенклатурность продукции предприятия, ее сложность и динамичность делают

задачу создания АСУ нереальной в условиях текущего на тот момент уровня технических и программных средств ЭВМ и тех плановых сроков, в рамках которых предполагалось решение поставленной задачи. Попытаемся экстраполировать это высказывание ученого на сферу IT, ДО и программирования. Мы допускаем, что существуют люди, для которых упомянутое высказывание не имеет никакого смысла и тем более практического значения. Главное, что должны понимать все участники разработки электронных пособий для системы ДО – это способность адекватно оценить все трудозатраты, связанные с проведением соответствующих работ. Развивая данную идею, позволим себе предположить, что способность четкой формализации проводимых работ, которую необходимо соблюдать как на уровне руководства программным проектом, так и на уровне непосредственных её исполнителей (программистов, преподавателей, методистов и т.д.) – это главное условие для достижения успеха [5], [6]. В подтверждение этого предположения мы приведем высказывание отца кибернетики Норберта Винера (1894 – 1964):

«...Едва ли кто-нибудь из нематематиков в состоянии освоиться с мыслью, что цифры могут представлять собой культурную и эстетическую ценность или иметь какое-нибудь отношение к таким понятиям, как красота, сила, вдохновение. Я решительно протестую против такого косного представления о математике. Существует немало математических работ... ничего не говорящих ни уму, ни сердцу. Но существуют и другие. Их авторы видят задачу математики в том, чтобы с помощью точных и четких методов создать новое, более совершенное представление о мире, высказать какое-то суждение, которое еще немного приоткроет завесу таинственности...». Способность во всем находить математику, пусть даже на самом её элементарном уровне – вот, что необходимо на настоящий момент. Мало кто задумывается еще над одной истиной, суть которой заключается в том, что любая программа, составленная человеком, при одних и тех же исходных данных будет выдавать одни и те же результаты. Исходя из этих предпосылок можно согласиться с тем, что возможен вариант написания практически любой программы. Единственное и достаточное условие, соблюдение которого приводит к истинности последнего высказывания – это опять-таки связано с математикой. Точнее её видение и поиск в рамках решения поставленной задачи. Это было доказано более 30 лет назад [2], когда мир узнал о том, что любую программу для компьютера можно представить при помощи всего лишь трёх вычислительных конструкций: условия, последовательность действий и циклические вычисления. При этом обязательно необходимо учитывать и общий уровень развития программного обеспечения в виде его совершенства и многофункциональности. В настоящее время, «…напротив, большинство наших систем гораздо сложнее, чем может считаться разумным, и слишком запутанны и хаотичны, чтобы ими было удобно и надежно пользоваться. Среднего пользователя компьютерной индустрии обслуживают так скверно, что он в любой момент ожидает сбоя своей системы, и мы наблюдали массовое распространение программного обеспечения, нашпигованного ошибками, по всему миру, за что нам должно было бы быть весьма стыдно…» [2].

Необходимость формализации основных этапов взаимодействия между всеми участниками разработки всякого программного обеспечения, в том числе и ДО, рассмотрим на примере построения динамических изображений с использованием фрактального подхода [7]. Опуская математические выкладки, вспомним, что построение фрактальных изображений основано на рекуррентной (итерационной) вычислительной процедуре, заданной функциональной зависимостью (*) и осуществляемой над точками, заданными в комплексной плоскости (**).

$$Z = F(Z) \quad – (*)$$

$$Z \in M – (**).$$

Примером может служить хорошо известное множество Мандельброта [7], формальная модель которой (*) представлена выражением Z [i+1] = Z [i] *Z[i] + Const, а область задания исходных точек определяется некоторой прямоугольной или другой областью в окрестностях начала координат.

На приведенном рис. 1 помещен скриншот уникального программного приложения, которое обеспечивает построение анимированного изображения в виде AVI файла, состоящего из серии последовательных кадров [8], каждый из которых специфицирует динамический переход от одной фрактальной модели к другой:

$$\begin{matrix} Z = FF(Z) & & G = FFF(G) \\ & \Rightarrow & \\ Z \in MM & & Z \in MMM \end{matrix}$$

Здесь: L – текущий номер кадра;
N_frame – общее число кадров;
ai, bi – действительные константы;

Sin(x)*Cos(y)	⇒	Sin(x)	} Начальное и конечное функциональные зависимости областей задания исходных точек.
Sin(y)*Cos(-x)		y	
2*(x1*x1 – y1*y1) + ai	⇒	(x1*x1 – y1*y1) + ai	} Начальная и конечная функции преобразования исходных точек.
4*x1*y1 + bi		2*x1*y1 + bi	

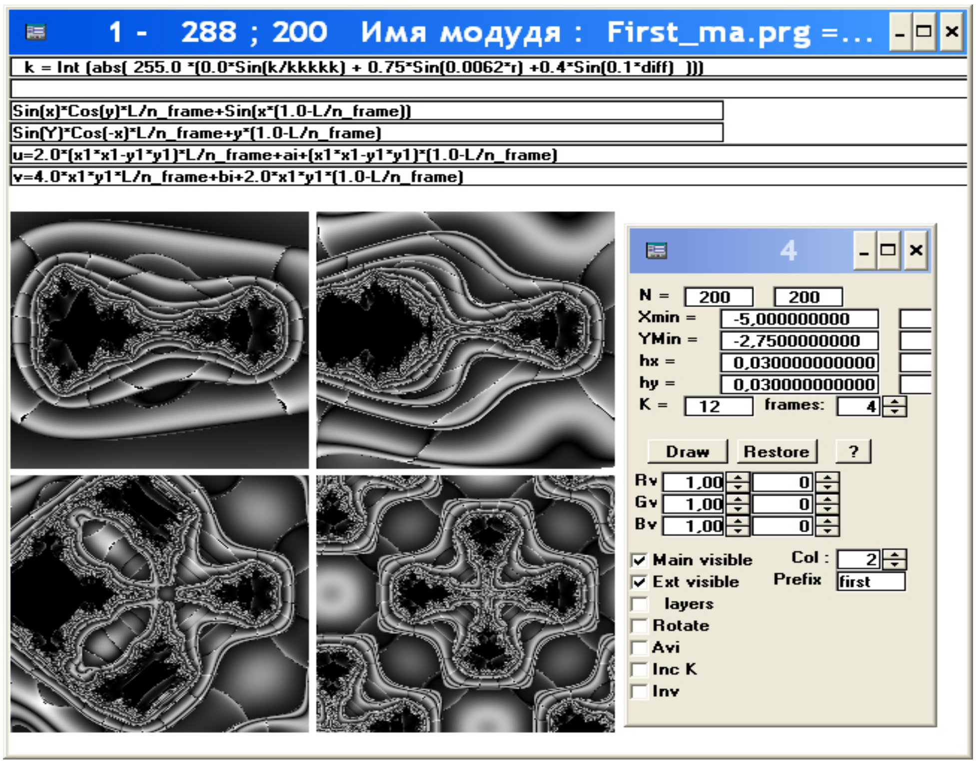


Рисунок 1 – Пример оригинальной математической модели для построения динамических фракталов

Из 4 помещенных кадров на рисунке – верхний левый специфицирует начальное состояние фрактального многообразия, а правый нижний – его конечное состояние.

Здесь может возникнуть вопрос относительно практической ценности предлагаемого подхода и реализованного на его основе программного продукта. Скептикам наш

ответ будет таким: «Старайтесь во всем видеть математику». Найдя её, реализуйте программу для компьютера, отладьте её и, если Вы получите те результаты, которые можно объяснить с высокой степенью достоверности можно предположить, что Вы на правильном пути.

Всякое программирование, как составляющая информатики, непременно предполагает очень глубокое понимание основных категорий и дефиниций, с которыми оно ассоциируется. Общеизвестно, что ООП держится на трех «китах» − наследование, инкапсуляция и полиморфизм. Информатика [9] также имеет своих «китов». Среди них, по нашему мнению, необходимо выделить такие понятия, как «алгоритм», «модель», «язык», «команда или оператор языка», «программа». Глубокое понимание сути этих понятий также носит аксиоматический характер. Особенно это актуально в наше время, когда Министерством образования и науки Украины анонсирована задача введения в практику обучения основ информатики и программирования с младших классов общеобразовательных учреждений.

Аксиомы программирования

Резюмируя все ранее изложенное и опираясь на тот весомый вклад, который внесли упомянутые здесь ученые, по нашему мнению, сам вопрос отнесения программирования к математическим наукам не вызывает никаких сомнений. Как и всякая другая строгая наука, она имеет свои основопологающие принципы (аксиомы), на базе которых и должно базироваться все то, что связано непосредственно как с программированием, так и с методами руководства коллективом, который занимается им. Знание и обязательное соблюдение этих аксиом позволит:

а) на базе строго сформулированного и однозначно интерпретируемого технического задания на разработку реально оценить длительность (сроки) проведения всего цикла проведения программных разработок;

а) минимизировать возможную задержку в плановых сроках выполнения программных разработок и их отдельных этапов;

б) практически исключить все возможные трения и непонимания в среде участников разработки, что в конечном итоге позволяет получение желаемое качество результирующего продукта.

Теперь сформулируем эти аксиомы.

Аксиома 1. Всякая программа, вне зависимости от того, в рамках какой языковой среды она реализована, при одних и тех же исходных и промежуточных данных на одном и том же оборудовании будет всегда выдавать одни и те же результаты.

Аксиома 2. Профессиональный программист может сконструировать любую программу: необходимым и достаточным условием этого является наличие формальной модели (алгоритма, блок-схема, техническое задание и т.д.), сконструированной в терминах трёх вычислительных конструкций: условия, последовательность действий и циклические вычисления.

Аксиома 3. Время разработки программного продукта объективно может быть оценено только тогда, когда у разработчика имеется формальная модель реализуемой им задачи.

Выводы

В данной работе впервые сформулированы аксиомы программирования, знание и соблюдение которых на практике позволяет разрешить ряд вопросов, связанных с

проектированием программного продукта . Общие вопросы, с поиском ответов на которые приходится всегда сталкиваться при такой разработке, могут быть с успехом использованы при проведении тендерных мероприятий, направленных, например, на подготовку электронных учебников или программно педагогических средств. В рамках проведения такого конкурса всем участникам может быть предложен вариант подготовки Betta-версии. Срок её реализации может быть ограничен 1 – 2 месяцами. На заключительном этапе конкурса выбор победителя самого тендера может быть объявлен не на основе вербальных пояснительных документов (техническое задание, технический проект), подготовленных в бумажном исполнении, а на базе тестирования (работы) предоставленных программных продуктов, пусть даже и в незавершенном, а в своём демонстрационном или частичном варианте. Именно такие условия проведения конкурса, во-первых, обеспечат наиболее эффективное использование материальных ресурсов, и значительно более короткие сроки исполнений тендерных заданий.

Литература

1. Научно-технический вестник СПб ГИТМО (ТУ). Выпуск 1. Новые образовательные технологии / Глав. ред. В.Н. Васильев. – СПб : СПб ГИТМО(ТУ), 2001. – 176 с.
2. Дейкстра Э. Программирование как вид человеческой деятельности [Электронный ресурс]. – Режим доступа : http://khpi-iip.mipk.kharkiv.edu/library/extent/dijkstra/
3. Скурихин В.И. О построении изображений некоторого класса объектов без алгоритмов удаления невидимых линий / В.И. Скурихин, М.А. Курилов // Искусственный интеллект. – 2005. – № 1. – С. 101-106.
4. Курилов М.А. Методология RAD в обучении / М.А. Курилов // Искусственный интеллект. – 2006. – № 1. – С. 47-56.
5. Иванова С.Б. Об одной модели обучения при построении «красивостей» / С.Б. Иванова, М.А. Курилов // Искусственный интеллект. – 2005. – № 3. – С. 494-499.
6. Иванова С.Б. Электронные учебники и средства их реализации / С.Б. Иванова, М.А. Курилов // Искусственный интеллект. – 2009. – № .4. – С. 344-348
7. Мандельброт Б. Фрактальная геометрия природы / Мандельброт Б. — М.: Институт компьютерных исследований, 2002.
8. Курилов М.А. Об одном подходе к построению динамических изображений на экране графического дисплея // Вопросы разработки математических методов и средств вычислительной техники. – Киев : ИК АН УССР, 1980. – С. 69-72.
9. Шишков Д.П. Философия компьютерной информатики (почему компьютеры принципиально не могут обрабатывать информацию) / Д.П. Шишков, А.И. Шевченко. – Донецк : ШПШІ, «Наука і освіта», 2009. – 252 с.

Literatura

1. Wikipedia. http://ru.wikipedia.org/wiki
2. Je.Dejkstra. Programmirovanie, kak vid chelovecheskojdejatel’osti. http://khpi-iip.mipk.kharkiv.edu/library/extent/dijkstra/
3. Skurihin V.I., Kurilov M.A. Iskusstvennyj intellekt. 2005. № 1. S.101-106.
4. Kurilov M.A. Iskusstvennyj intellekt. 2006. № 1. S. 47-56.
5. Ivanova S. B., Kurilov M. A. Iskusstvennyj intellekt. 2005. № 3. S. 494-499.
6. Ivanova S.B., Kurilov M.A. Iskusstvennyj intellekt. 2009. № 4. S. 344-348.
7. Mandel’brot B. Fraktal’naja geometrija prirody. M.: “Institut komp’juternyh issledovanij”. 2002.
8. Kurilov M.A. Voprosy razrabotki matematicheskih metodov i sredstv vychislitel'noj tehniki. Kiev: IK AN USSR. 1980. S. 69-72.
9. Shishkov D.P., Shevchenko A.I. Filosofija komp’juternoj informatiki (pochemu komp’jutery principial’no ne mogut obrabatyvat’ informaciju). Doneck: ShPShI. “Nauka i osvita”. 2009. 252s

A.I. Shevchenko, M.A. Kurilov, L.P. Sypchenko, A.S. Barashko

Axioms of Programming and Some Issues of Distance Education

In the given work, axioms of programming are offered, that is the novelty. These axioms help in quick and qualitative product development. The fact that programming pertains to mathematical science does not raise doubts beside specialist in the field of information technology. Practical practicability of consideration of the question about teaching programming under point of the mathematician vision is persuasively proved on the example of one problem.

It is known that building of fractal scenes is founded on recursive (iteration) computing procedure given by functional dependency (*) and realized on point, given in complex plane (**).

$$Z = F(Z) \quad - (*)$$
$$Z \quad M \quad - (**)$$

The example can be well known Mandelbrot set. Unlike traditional (steady-state) use of this device, in work there are given the results of the work of a program model, with the help of which dynamic transition from one complex variety to another one has been realized. Here with, the domains of definition for these varieties also fall under the dynamic transformation.

$Z = FF(Z)$ ⇒ $G = FFF(G)$

$Z \in MM$ ⇒ $Z \quad MMM$

Sin(x)*Cos(y) ⇒ Sin(x)

Sin(y)*Cos(-x) ⇒ y

Initial and final functional dependencies of domains of definition for source point.

2*(x1*x1 - y1*y1) + ai ⇒ (x1*x1 - y1*y1) + ai

4*x1*y1 + bi ⇒ 2*x1*y1 + bi

Initial and final transform functioning for source point.

The example of the other practical application of the offered device can be a variant of the undertaking tender action directed, for instance, on preparation of electronic textbooks or software facilities for pedagogical needs. Within the contest, a variant of preparation of Betta-version can be offered to all participants. The term for the realization can be limited by 1-2 months. At the final stage of the contest, tender award can be declared not on the base of word explanatory documents (the technical requirement, detail design) prepared in papers, but on the base of the testing (work) of the given program products, even if they are not finished yet but in their demonstration or partial variant. Namely such condition for the contest will provide the most efficient use of material resource and much shorter periods for doing tender tasks.

Статья поступила в редакцию 02.12.2011.